Abstract

Networks that use SIP to start and stop sessions between their users will frequently be upgraded with software and hardware changes. Users will similarly frequently change their client software and the way they use the network. In order to allow troubleshooting and regression testing, it is useful to provide debugging as part of the network fabric. This draft describes a SIP private header that triggers logging of SIP signalling and identifies logs at multiple SIP entities as belonging to a single end-to-end session.
1. Introduction

Alice has a SIP client on her laptop, which she has been using for some time to make video calls to work colleagues inside her company. Today, she tried to set up a call to Bob, who recently installed an audio-only SIP phone at home, but the call failed and Alice does not know why. She contacts those who manage the SIP network within her company to ask them to fix the problem.

Alice’s UA and the SIP proxy that Alice uses must be configured to log SIP signalling the next time she sends an INVITE request. The UA and the proxy obtain their configuration by subscribing to the debug event package, which supplies XML configuration documents carried in NOTIFY requests. Because debugging is rarely needed, the debug event package should only be subscribed to when required, which is achieved by triggering subscription when Alice refreshes her registration. The administrators cause Alice to re-register by notifying her UA that its subscription has expired. When Alice’s UA re-registers, an empty P-Debug-ID header field is included in the 200 OK response to the REGISTER request. This empty P-Debug-ID header field causes both Alice’s UA and the SIP proxy that Alice uses to subscribe to Alice’s debug event package at the registrar, which returns them an XML document containing her debugging configuration.

The debugging configuration causes Alice’s UA and the SIP proxy that Alice uses to log SIP signalling the next time she sends an INVITE request. Alice retries calling Bob and signalling is logged until the dialog with Bob ends. Later examination of these logs shows that although requests and responses are correctly exchanged with Bob, Alice’s SIP client is not accepting audio-only sessions and is sending BYE immediately. This problem had not come to light previously as all calls within Alice’s company are video calls.

The debugging configuration (supplied by subscription to the debug event package) used to investigate the problem is shown below. Alice’s UA inserts the configured identifier (P-Debug-ID:A076D1) to trigger logging of signalling at the proxy, and the same identifier is used to correlate signalling logged at Alice’s UA and at the Proxy after logging has finished.
Figure 1: Debugging Configuration

The outline call flow below illustrates how debugging works. Signalling logged at Alice’s UA and the Proxy shows that requests and responses are successfully exchanged, but Alice’s UA will not set up an audio-only session and sends BYE immediately.
Alice                        Proxy                        Bob

| (1) INVITE                      |                       |
| m = audio                       |                       |
| From: alice at atlanta.com      |                       |
| P-Debug-ID: A076D1              |                       |
| Alice’s UA starts logging       |                       |
-----------------------------

| (2) INVITE                      |                       |
| P-Debug-ID: and From:           |                       |
| match debugging config          |                       |
| so proxy starts logging         |                       |
-----------------------------

| (3) 200 OK                      |                       |
| m = audio                       |                       |
<-----------------------------

| (4) 200 OK                      |                       |
<-----------------------------

| (5) ACK                         |                       |
-----------------------------

| (6) ACK                         |                       |
-----------------------------

| (7) BYE                         |                       |
-----------------------------

| (8) BYE                         |                       |
-----------------------------

| (9) 200 OK                      |                       |
<-----------------------------

| (10) 200 OK                     |                       |
<-----------------------------

Dialog has ended so Proxy stops logging

Figure 2: Example of Debugging

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
2. The P-Debug-ID Header

The P-Debug-ID header field carries a 3-digit hexadecimal number used as an end-to-end identifier for logging of SIP signalling. The header is added by a SIP UA or a SIP proxy in order to trigger logging of SIP signalling in downstream entities. Configuration of debugging is provided by the debug event package described in draft-dawes-sipping-debug-event [draft-dawes-sipping-debug-event]

2.1. Debugging principle of operation

Debugging can be understood by the simple state machine below, which applies to any SIP UA or Proxy.

```
+------------+
|            |
| Inactive   |
+------------+
   ^
   | Delete debugging configuration
   V
+------------+
|            |
| Active     |
+------------+
   ^
   | Supply debugging configuration
   V
+------------+
|            |
| Logging    |
+------------+
```

Figure 3: Debugging State Machine

This document is mainly concerned with the "Start trigger event"; debugging configuration and the stop trigger event are described to illustrate the purpose of the P-Debug-ID header.
2.2. Role of P-Debug-ID

P-Debug-ID has three roles, to provide the start trigger event in the figure above, to cause (by its presence) a SIP entity to log SIP signalling, or to cause a UA to subscribe to the debug event package described in draft-dawes-sipping-debug-event.

P-Debug-ID provides the start trigger event in the example in the figure above for all SIP entities other than the SIP entity that inserted the P-Debug-ID header. Start of logging of SIP signalling by a SIP entity is triggered if the P-Debug-ID header contains a value that matches the value contained in the debugging configuration of that SIP entity. Logging continues until a "stop trigger event" is detected, as defined within the configuration element "stop trigger event".

Once a non-empty P-Debug-ID header field has been inserted, SIP entities that do not have any debugging configuration can use its presence as an indication that signalling MUST be logged.

Debugging will be an infrequent activity, therefore it is not efficient for a UA to be permanently subscribed to the debug event package. A registrar can prompt a UA to subscribe to the debug event package by including an empty P-Debug-ID header field in a 200 OK response to a REGISTER request.

2.3. Detecting when to insert P-Debug-ID

A UA, proxy, or registrar can insert a P-Debug-ID header. Debugging configuration MAY specify conditions that must be met for to insert a P-Debug-ID header in the <start-trigger> configuration element. Typically, the conditions will be a particular SIP method and a particular SIP URI in the From: header field, for UA originating requests, or the To: header field, for UA terminating requests. For example, a UA that originates an INVITE request and identifies itself as alice@u1.atlanta.com will trigger that UA to insert a P-Debug-ID header containing a 3-digit hexadecimal value taken from the <debug-id> configuration element.

2.4. Identifying logged signalling across entities

The P-Debug-ID header field contains a 6-digit hexadecimal number, taken from the <debug-id> configuration element, which is combined with the address of record attribute of the "aor" attribute in configuration data to provide a unique identifier to tie together SIP signalling logged at all UAs and proxys. Logging of SIP signalling must include the value in the P-Debug-ID header field and the user
identity from the <user-id> configuration element.

2.5. Validity of debugging configuration

Debugging configuration is used once and then discarded. Debugging configuration is valid until the following sequence has completed: a start trigger event defined in the <start-trigger> configuration element is detected, logging of signalling starts, the stop trigger event defined in the <stop-trigger> configuration element is detected, and logging of signalling stops. The configuration is then discarded. If no debugging configuration remains, the entity moves into the Inactive state in the debugging state machine.

2.6. Usage of the P-Debug-ID header

A UA, proxy, or registrar can insert a P-Debug-ID header. P-Debug-ID has three roles: to provide the start trigger event in the figure above, to cause an entity with no debugging configuration to log a message, or to cause a UA to subscribe to the debug event package described in draft-dawes-sipping-debug-event [draft-dawes-sipping-debug-event].

2.6.1. Procedures at the UA

If the UA is originating a SIP session and detects a start trigger event, as defined in its debugging configuration information, and is not logging SIP signalling, the UA MUST insert a P-Debug-ID header field containing the 3-digit hexadecimal value defined in the <debug-id> sub-element of its debugging configuration.

If the UA is terminating a SIP session and detects a start trigger event, as defined in its debugging configuration information, the UA MUST begin to log SIP signalling.

If the UA is logging SIP signalling and detects a stop trigger event, as defined in its debugging configuration information, the UA MUST stop logging SIP signalling.

A UA MUST copy the P-Debug-ID header from a terminating request into all responses to that request, including 1xx responses.

If a UA receives an empty P-Debug-ID header field in a 200 OK response to a REGISTER request, the UA SHOULD subscribe its own debug event package, defined in draft-dawes-sipping-debug-event [draft-dawes-sipping-debug-event], using the address of record that it registered.
2.6.2. Procedures at the Registrar

If a registrar detects a start trigger event, as defined in its debugging configuration information, and is not logging SIP signalling, the registrar SHOULD begin to log SIP signalling.

If the SIP session is terminating at a UA served by the registrar and the registrar detects a start trigger event, as defined in its debugging configuration information and the SIP request does not contain a P-Debug-ID header and a 3-digit hexadecimal value is defined in the <debug-id> sub-element of its debugging configuration then the registrar MUST insert a P-Debug-ID header field containing the 3-digit hexadecimal value defined in the <debug-id> sub-element of its debugging configuration.

If the registrar detects a stop trigger event, as defined in its debugging configuration information, the registrar SHOULD stop logging SIP signalling.

If the registrar forks a SIP request, the registrar MUST copy the P-Debug-ID header field into each request that results from forking.

A registrar MUST copy the P-Debug-ID header from a request into all responses to that request, including 1xx responses.

The registrar MAY include an empty P-Debug-ID header in a 200 OK response to a REGISTER request to prompt a UA to subscribe to the debug event package described in draft-dawes-sipping-debug-event [draft-dawes-sipping-debug-event].

2.6.3. Procedures at a Proxy

2.6.3.1. Proxy with Debug Configuration

If a proxy detects a start trigger event, as defined in its debugging configuration information, and is not logging SIP signalling, the registrar MUST begin to log SIP signalling.

If the SIP session is originating at a UA served by the proxy and the proxy detects a start trigger event, as defined in its debugging configuration, and the SIP request does not contain a P-Debug-ID header and a 3-digit hexadecimal value is defined in the <debug-id> sub-element of its debugging configuration then the proxy MUST insert a P-Debug-ID header field containing the 6-digit hexadecimal value defined in the <debug-id> sub-element of its debugging configuration. If the proxy received the message from an element that it does not trust and there is a P-Debug-ID header present containing a 3-digit hexadecimal value, the proxy MUST replace that value with the value
defined in the <debug-id> sub-element of its debugging configuration or remove this header field.

If the proxy detects a stop trigger event, as defined in its debugging configuration, the registrar SHOULD stop logging SIP signalling.

A proxy MUST copy the P-Debug-ID header from a request into all responses to that request, including 1xx responses.

2.6.3.2. Proxy without Debug Configuration

A proxy in a Trust Domain can receive a request from a node that it trusts, or a node that it does not trust. If the proxy receives a message (request or response) from a node that it trusts, it can use the presence of the P-Debug-ID header field, if any, as an indication that the message MUST be logged.

2.7. Example Call Flow

In order to trigger logging, each entity must be pre-configured to allow it to detect start and stop trigger events.

2.7.1. Example configuration at the user agent

In this example, the UA is configured to start logging at 9am by the <time> sub-element of the <start-trigger> element. The <stop-trigger> element causes logging to stop 6 minutes after it starts. The <debug-id> sub-element is part of configuration data and provides the 6-digit hexadecimal identifier that the UA will include in the P-Debug-ID header field when it next sends a SIP request.

```xml
<?xml version="1.0"?>
<debuginfo xmlns="urn:ietf:params:xml:ns:debuginfo" version="0" state="full">
  <debugconfig aor="alice@u1.atlanta.com">
    <session id="r00">
      <start-trigger>
        <time>09:00:00</time>
      </start-trigger>
      <stop-trigger>
        <time-period>T0H6M0S</time-period>
      </stop-trigger>
      <debug-control>
        <debug-id>1A346D</debug-id>
      </debug-control>
    </session>
  </debugconfig>
</debuginfo>
```
2.7.2. Example configuration at a proxy

The configuration at the proxy causes the proxy to start logging when it receives the value 1A346D in the P-Debug-ID header field from user alice@u1.atlanta.com. The proxy will stop logging signalling 6 minutes after it starts, as indicated by the <time-period> sub-element of the <stop-trigger> element.

```xml
<?xml version="1.0"?>
<debuginfo xmlns="urn:ietf:params:xml:ns:debuginfo" version="0" state="full">
  <debugconfig aor="alice@atlanta.com">
    <session id="r01">
      <from>alice@atlanta.com</from>
      <start-trigger>
        <debug-id>1A346D</debug-id>
      </start-trigger>
      <stop-trigger>
        <time-period>T0H6M0S</time-period>
      </stop-trigger>
      <debug-control>
        <depth>minimum</depth>
      </debug-control>
    </session>
  </debugconfig>
</debuginfo>
```

2.7.3. Triggering logging start and stop

Starting at 9 am, the UA starts to log any SIP signalling and continues logging for 6 minutes. During this period, the UA originates an INVITE request that passes through proxy p1.atlanta.com

```
User                    Proxy
u1.atlanta.com          p1.atlanta.com
<p>| |
|                         |
| (1) INVITE              |                          |</p>
<table>
<thead>
<tr>
<th>P-Debug-ID:1A346D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(2) INVITE</td>
</tr>
<tr>
<td></td>
<td>P-Debug-ID:1A346D</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</table>
```
When proxy p1.atlanta.com receives the INVITE request, it examines the P-Debug-ID header and the user identity in the From: header field and find that the values match the <debug-id> and <from> configuration elements. The proxy therefore begins to log signalling for the next 6 minutes, as configured in the <time-period> configuration element.

Alice sends an INVITE request that is tagged for debugging (1):

INVITE sip:alice@atlanta.com SIP/2.0
Via: SIP/2.0/UDP u1.atlanta.com;branch=z9hG4bKnashds8
From: sip:alice@u1.atlanta.com;tag=123aa10
To: sip:bob@biloxi.com
P-Preferred-Identity: alice@u1.atlanta.com
Call-ID: 9901@u1.atlanta.com
CSeq: 82779 INVITE
Max-Forwards: 70
P-Debug-ID: 1A346D
Content-Type: application/sdp
Content-Length: ...

The Proxy compares the value in the P-Debug-ID of the INVITE request and the user identity in the From: header field with its debug configuration and, if a match is found, begins to log signalling. The proxy propagates the INVITE request onwards (2):

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP u1.atlanta.com;branch=z9hG4bKnashds8
Via: SIP/2.0/UDP p1.atlanta.com;branch=z9hG4bKnashds8
From: sip:alice@atlanta.com;tag=123aa10
To: sip:bob@biloxi.com
P-Asserted-Identity: alice@atlanta.com
Call-ID: 9901@u1.atlanta.com
CSeq: 82779 INVITE
Max-Forwards: 69
P-Debug-ID: 1A346D
Content-Type: application/sdp
Content-Length: ...
3. Formal Syntax

All of the mechanisms specified in this document are described in both prose and an augmented Backus-Naur Form (BNF) defined in RFC 2234 [RFC2234]. Further, several BNF definitions are inherited from SIP and are not repeated here. Implementors need to be familiar with the notation and contents of SIP RFC 3261 [RFC3261] and RFC 2234 [RFC2234] to understand this document.

3.1. P-Debug-ID header syntax

The syntax for the P-Debug-ID header is described as follows:

P-Debug-ID = "P-Debug-ID" HCOLON gen-value

4. Table of new headers

Table 1 extends the headers defined in this document to Table 2 in SIP RFC 3261 [RFC3261], section 7.1 of the SIP-specific event notification RFC 3265 [RFC3265], tables 1 and 2 in the SIP INFO method RFC 2976 [RFC2976], tables 1 and 2 in Reliability of provisional responses in SIP RFC 3262 [RFC3262], tables 1 and 2 in the SIP UPDATE method RFC 3311 [RFC3311], tables 1 and 2 in the SIP extension for Instant Messaging RFC 3428 [RFC3428], and table 1 in the SIP REFER method RFC 3515 [RFC3515]:

<table>
<thead>
<tr>
<th>Header field</th>
<th>where</th>
<th>proxy</th>
<th>ACK</th>
<th>BYE</th>
<th>CAN</th>
<th>INV</th>
<th>OPT</th>
<th>REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-Debug-ID</td>
<td>car</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

Table 1: Header field support

5. Acknowledgements

This template was derived from an initial version written by Pekka Savola and contributed by him to the xml2rfc project.
6. IANA Considerations

This document defines one private SIP extension header field (beginning with the prefix "P-"").

This extension header will be included in the registry of SIP header fields defined in SIP RFC 3261 [RFC3261]. Expert review as required for this process will be requested from the SIP Working Group.

The following extension is to be registered as a private extension header field:

- RFC Number: RFC????
- Header Field Name: P-Debug-ID
- Compact Form: none

7. Security Considerations

The identity carried by the P-Debug-ID header is not sensitive information, although it will sometimes indicate that a particular device is experiencing problems. If the value in the header is maliciously changed, this will disrupt troubleshooting.

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Additional Stuff

This becomes an Appendix.
Authors’ Addresses

Peter Dawes
Vodafone Group
The Connection
Newbury, Berkshire  RG14 2FN
UK

Phone: +44 7717 275009
Email: peter.dawes@vodafone.com

Kar Ann Chew (editor)
Vodafone Group
The Connection
Newbury, Berkshire  RG14 2FN
UK

Phone:
Email: karann.chew@vodafone.com
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.